TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI
Khoa Công Nghệ Thông Tin

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI
Xây dựng hệ thống tóm tắt tin tức tiếng Việt tự động với mô hình ViT5

Sinh viên thực hiện:	Nguyễn Trung Kiên
Mã số sinh viên:	0093267
Lớp:	67CNCS
Chuyên ngành:	Khoa học máy tính
Giảng viên hướng dẫn:	TS. Phạm Hồng Phong

Hà Nội, 12–2025

Mục lục

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

TÓM TẮT NỘI DUNG ĐỒ ÁN

Với sự bùng nổ thông tin trên các trang báo điện tử, việc nắm bắt nhanh nội dung tin tức trở nên cần thiết hơn bao giờ hết. Nhiều hệ thống tóm tắt đã được phát triển cho tiếng Anh, nhưng đa phần thiếu khả năng xử lý ngữ cảnh tiếng Việt phức tạp hoặc yêu cầu tài nguyên tính toán lớn. Các phương pháp extractive truyền thống dựa trên TF-IDF hoặc TextRank cho kết quả nhanh nhưng thiếu tính tự nhiên, trong khi abstractive thường phụ thuộc vào dataset có chất lượng cao và khó thu thập.

Đồ án lựa chọn triển khai hai hướng tiếp cận độc lập để đánh giá toàn diện khả năng tóm tắt tin tức tiếng Việt. Phương pháp thứ nhất là **extractive** dựa trên TF-IDF và K-Means clustering để tự động tạo dataset từ 11,353 bài báo. Hướng thứ hai sử dụng **abstractive** với dataset mở chứa tóm tắt viết tay chất lượng cao, huấn luyện mô hình paraphrase nội dung. Cả hai đều sử dụng kiến trúc ViT5-base (220M parameters) nhưng với cấu hình khác nhau phù hợp đặc thú từng phương pháp. Việc áp dụng đồng thời hai hướng giúp tận dụng ưu điểm từng mô hình và cho phép đối sánh khách quan.

Giải pháp tổng quan bao gồm xây dựng pipeline crawl dữ liệu từ các nguồn báo điện tử, xử lý và làm sạch văn bản, tạo dataset training tự động (extractive) và thủ công (abstractive), huấn luyện 2 mô hình ViT5 độc lập, đánh giá với ROUGE và BERT F1, và tích hợp hệ thống web với streaming real-time. Đóng góp chính của đồ án là xây dựng và so sánh hai phương pháp tóm tắt một cách hệ thống, làm rõ ưu – nhược điểm và khả năng ứng dụng của từng hướng. Kết quả đánh giá cho thấy extractive đạt kết quả khá tốt nhưng thời gian inference chậm; trong khi abstractive vừa đạt kết quả tốt vừa tạo tóm tắt ngắn gọn hơn và nhanh gấp đôi. Hệ thống web đã triển khai thành công với FastAPI backend, React frontend, hỗ trợ crawl streaming và category extraction tự động.

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

Lời cảm ơn

Trong suốt thời gian học tập và thực hiện đồ án tốt nghiệp tại Trường Đại học Xây dựng Hà Nội, em đã nhận được rất nhiều sự giúp đỡ và hỗ trợ quý báu từ phía Nhà trường, các thầy cô, gia đình và bạn bè.

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc tới **TS. Phạm Hồng Phong**, giảng viên Khoa Công nghệ Thông tin, người đã tận tình hướng dẫn, định hướng về mặt chuyên môn cũng như góp ý chi tiết trong suốt quá trình em thực hiện đề tài. Những ý kiến chỉ dẫn của thầy đã giúp em hoàn thiện từ ý tưởng, mô hình, cách xây dựng pipeline đến cách trình bày và đánh giá hệ thống.

Em xin chân thành cảm ơn các thầy cô trong **Khoa Công nghệ Thông tin** đã truyền đạt cho tôi nền tảng kiến thức vững chắc về khoa học máy tính, học máy, xử lý ngôn ngữ tự nhiên và phát triển ứng dụng, là cơ sở quan trọng để em có thể triển khai đề tài này.

Em cũng xin gửi lời cảm ơn tới gia đình và bạn bè đã luôn động viên, khích lệ và tạo điều kiện thuận lợi cho em trong suốt quá trình học tập và làm đồ án.

Mặc dù đã cố gắng hết sức, báo cáo và hệ thống chắc chắn không tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp của các thầy cô để có thể hoàn thiện hơn trong các nghiên cứu, công việc sau này.

Hà Nội, tháng 12 năm 2025

Nguyễn Trung Kiên

Chương 1: Giới thiệu

1.1 Lý do chọn đề tài

Trong bối cảnh bùng nổ thông tin hiện nay, người dùng Internet Việt Nam tiếp cận một lượng tin tức khổng lồ mỗi ngày từ nhiều báo điện tử. Các bài báo thường dài, phức tạp, khiến việc nắm bắt nhanh thông tin trở nên khó khăn. Việc tận dụng các kỹ thuật xử lý ngôn ngữ tự nhiên và mô hình học sâu để tự động tóm tắt tin tức có thể giúp người dùng tiết kiệm thời gian nhưng vẫn đảm bảo chất lượng tiếp nhận thông tin.

1.2 Mục tiêu của đề tài

Đề tài hướng tới xây dựng một hệ thống đọc tin nhanh tiếng Việt với các mục tiêu cụ thể:

- Thu thập và xử lý tin tức từ các nguồn báo điện tử trong nước (VnExpress, Vietnamnet).
- Xây dựng hai mô hình tóm tắt tin tức dựa trên ViT5:
  - Mô hình huấn luyện trên dữ liệu extractive tự thu thập ($\sim$15,000 mẫu).
  - Mô hình huấn luyện trên dữ liệu abstractive công khai từ HuggingFace.
- Tự động phân loại tin tức theo 11 chủ đề dựa trên cấu trúc URL.
- Xây dựng giao diện web (FastAPI + React) cho phép xem, lọc và cập nhật tin tức.
- So sánh và đánh giá chất lượng hai mô hình trên các chỉ số ROUGE.

1.3 Đối tượng và phạm vi nghiên cứu

Đối tượng: Văn bản tin tức tiếng Việt, các kỹ thuật tóm tắt văn bản dựa trên mô hình Transformer (T5), và kiến trúc hệ thống xử lý tin tức thời gian thực.

Phạm vi:

- Nguồn dữ liệu: VnExpress và Vietnamnet.[1, 2]
- Chỉ xử lý văn bản (không xử lý ảnh, video).

- Hai phương pháp tạo dữ liệu huấn luyện: extractive (TF-IDF + KMeans) và abstractive (dataset công khai).
- Phân loại 11 chủ đề dựa trên URL: Chính trị, Thế giới, Kinh doanh, Khoa học, Giải trí, Thể thao, Pháp luật, Giáo dục, Sức khỏe, Đời sống, Du lịch.

1.4 Phương pháp nghiên cứu và hướng tiếp cận

Đề tài áp dụng các phương pháp:

- **Nghiên cứu lý thuyết:** Tìm hiểu các mô hình Transformer, T5/ViT5 và kỹ thuật tóm tắt văn bản.
- **Thực nghiệm:**
  - Thu thập 15,000 bài báo, sinh nhãn extractive bằng TF-IDF + KMeans.
  - Sử dụng dataset abstractive "8Opt/vietnamese-summarization-dataset-0001".
  - Huấn luyện và đánh giá hai mô hình ViT5-base độc lập.
- **Hiện thực hóa:** Xây dựng pipeline crawl $\rightarrow$ tóm tắt $\rightarrow$ phân loại URL $\rightarrow$ hiển thị web.

Hướng tiếp cận chính là so sánh hai cách tạo dữ liệu huấn luyện: extractive (domain-specific, nhãn tự động) và abstractive (chất lượng cao, general domain), từ đó đánh giá mô hình nào phù hợp hơn cho bài toán tóm tắt tin tức Việt Nam.

Chương 2: Cơ sở lý thuyết và công nghệ

2.1 Tóm tắt văn bản

Tóm tắt văn bản (text summarization) là bài toán tự động tạo ra một đoạn văn bản ngắn gọn nhưng vẫn bao quát được nội dung chính của văn bản gốc. Trong lĩnh vực xử lý ngôn ngữ tự nhiên, có hai hướng tiếp cận chính được phân biệt dựa trên cách thức sinh tóm tắt:

- **Tóm tắt trích rút (extractive summarization)**: Lựa chọn và ghép lại các câu quan trọng có sẵn từ văn bản gốc. Phương pháp này đảm bảo tính chính xác về mặt ngữ pháp và giữ nguyên các thông tin gốc, nhưng đôi khi thiếu sự tự nhiên và mạch lạc do các câu được lấy từ nhiều vị trí khác nhau.
- **Tóm tắt trừu tượng (abstractive summarization)**: Mô hình sinh ra các câu mới dựa trên việc hiểu ngữ nghĩa của văn bản gốc. Hướng tiếp cận này cho phép tạo ra tóm tắt tự nhiên, súc tích và linh hoạt hơn, tuy nhiên đòi hỏi kiến trúc mô hình phức tạp và có thể gây ra hiện tượng "ảo giác" (hallucination) - sinh ra thông tin không có trong văn bản gốc.

Đề tài này tập trung vào hướng tiếp cận abstractive bằng mô hình ViT5, đồng thời so sánh với phương pháp extractive dựa trên TF-IDF và K-Means clustering để đánh giá hiệu quả của từng hướng.

2.2 Kiến trúc Transformer

2.2.1 Tổng quan

Transformer [3] được giới thiệu bởi Vaswani và cộng sự năm 2017 trong bài báo "Attention Is All You Need". Kiến trúc này đã thay thế hoàn toàn các mạng tuần tự truyền thống như RNN và LSTM bằng cơ chế self-attention, cho phép xử lý song song toàn bộ chuỗi đầu vào và học được các quan hệ phụ thuộc dài hạn tốt hơn.

Các ưu điểm nổi bật của Transformer bao gồm:

- Khả năng song song hóa cao, tận dụng hiệu quả GPU/TPU
- Học được quan hệ giữa các token ở xa nhau trong chuỗi
- Không bị vấn đề vanishing gradient như RNN
- Có thể mở rộng quy mô lên hàng tỷ tham số

2.2.2 Cơ chế Self-Attention

Self-attention là cơ chế cốt lõi của Transformer, cho phép mô hình đánh giá mức độ liên quan giữa mọi cặp token trong chuỗi. Với mỗi token, mô hình sẽ tạo ra ba vector đại diện:

- **Query (Q)**: Vector "câu hỏi" của token hiện tại, thể hiện token đang tìm kiếm thông tin gì
- **Key (K)**: Vector đại diện cho mỗi token, dùng để đối chiếu với Query
- **Value (V)**: Vector chứa thông tin thực tế của token

Công thức tính attention được định nghĩa như sau:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

trong đó d_k là chiều của vector Key. Việc chia cho $\sqrt{d_k}$ (scaled dot-product attention) giúp chuẩn hóa các giá trị attention score và ổn định quá trình huấn luyện, tránh hiện tượng gradient quá nhỏ khi d_k lớn.

Multi-Head Attention mở rộng cơ chế này bằng cách chạy song song nhiều attention heads, mỗi head học các mẫu quan hệ khác nhau giữa các token:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_h)W^O$$

2.2.3 Kiến trúc Encoder-Decoder

Transformer nguyên bản sử dụng kiến trúc encoder-decoder, phù hợp cho các bài toán sequence-to-sequence như dịch máy và tóm tắt văn bản:

- **Encoder**: Xử lý chuỗi đầu vào song song qua các lớp self-attention và feed-forward network. Mỗi lớp encoder tạo ra biểu diễn ngữ nghĩa ngày càng trừu tượng của văn bản đầu vào.
- **Decoder**: Sinh chuỗi đầu ra theo kiểu tự hồi quy (auto-regressive), tức mỗi token được sinh dựa trên các token đã sinh trước đó. Decoder sử dụng:
  - Masked self-attention: chỉ nhìn thấy các token đã sinh
  - Cross-attention: tập trung vào các phần liên quan trong encoder output

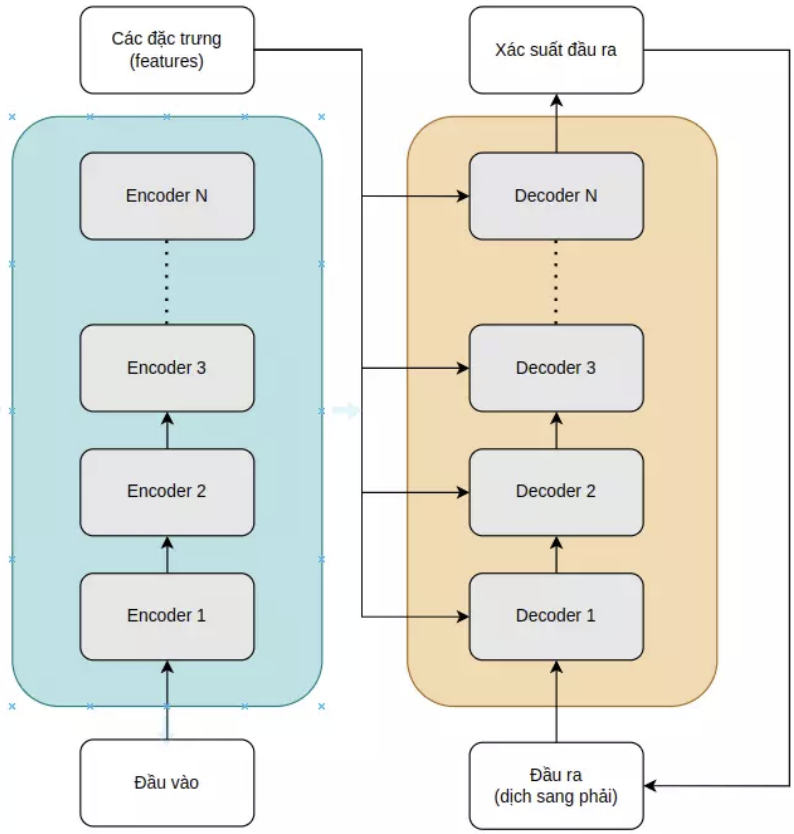

Figure 2.1: Kiến trúc Transformer encoder-decoder

2.3 Mô hình T5 và ViT5

2.3.1 T5 - Text-to-Text Transfer Transformer

T5 (Text-to-Text Transfer Transformer) [4] do Google Research giới thiệu năm 2019, đã thống nhất mọi tác vụ xử lý ngôn ngữ tự nhiên thành dạng text-to-text. Thay vì thiết kế kiến trúc riêng cho từng bài toán, T5 sử dụng cùng một kiến trúc encoder-decoder cho mọi tác vụ bằng cách thêm prefix vào đầu vào:

- Tóm tắt: `"summarize: [văn bản]"` → `"[tóm tắt]"`
- Dịch: `"translate English to Vietnamese: Hello"` → `"Xin chào"`
- Phân loại: `"sentiment: This is great!"` → `"positive"`

T5 được pre-train trên Colossal Clean Crawled Corpus (C4) - một bộ dữ liệu tiếng Anh khổng lồ khoảng 750GB, sử dụng objective là *span corruption*: che ngẫu nhiên các đoạn văn bản liên tiếp và yêu cầu mô hình phục hồi các đoạn bị che đó.

2.3.2 ViT5 - Vietnamese T5

ViT5 [5] là phiên bản T5 được VietAI phát triển riêng cho tiếng Việt vào năm 2022. Do T5 gốc được huấn luyện chủ yếu trên tiếng Anh, việc áp dụng trực tiếp cho tiếng Việt gặp nhiều hạn chế về tokenization và hiểu ngữ nghĩa. ViT5 khắc phục vấn đề này bằng cách:

- Huấn luyện lại tokenizer SentencePiece từ đầu trên corpus tiếng Việt
- Xây dựng vocabulary 32,000 tokens phù hợp với đặc thù từ đa âm tiết của tiếng Việt
- Pre-training lại toàn bộ mô hình trên khoảng 60GB văn bản tiếng Việt từ báo chí, Wikipedia, sách và các nguồn web

Đề tài này sử dụng `VietAI/vit5-base` làm mô hình nền và huấn luyện hai mô hình độc lập cho hai hướng tiếp cận khác nhau:

Hai hướng tiếp cận:

- **Mô hình Extractive**: Fine-tune ViT5 trên tóm tắt được tạo tự động bằng phương pháp TF-IDF và K-Means clustering. Dữ liệu huấn luyện là các bài báo tiếng Việt được thu thập và tóm tắt tự động bằng thuật toán extractive. Độ dài đầu vào và đầu ra lớn hơn (1500/320 tokens) để phù hợp với tóm tắt extractive thường dài hơn.
- **Mô hình Abstractive**: Fine-tune ViT5 trên dataset tóm tắt abstractive chuẩn [6], trong đó các tóm tắt được viết bởi con người (sa-pô báo chí). Độ dài ngắn hơn (1280/256 tokens) do tóm tắt abstractive thường ngắn gọn và súc tích hơn.

Hai mô hình này hoàn toàn độc lập và được huấn luyện từ cùng một checkpoint `VietAI/vit5-base`, cho phép so sánh trực tiếp hiệu quả của hai phương pháp tóm tắt trên cùng một kiến trúc nền.

2.4 Phương pháp tóm tắt trích rút với TF-IDF và K-Means

Để so sánh với phương pháp trừu tượng, đề tài cũng triển khai một pipeline tóm tắt trích rút dựa trên học máy truyền thống. Phương pháp này không sử dụng mô hình ngôn ngữ lớn mà dựa vào các kỹ thuật thống kê và clustering.

2.4.1 TF-IDF (Term Frequency - Inverse Document Frequency)

TF-IDF là phương pháp biểu diễn văn bản dựa trên tần suất từ, trong đó mỗi câu được chuyển thành một vector số:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t)$$

$$\text{IDF}(t) = \log \frac{N}{\text{df}(t)}$$

trong đó:

- t là một từ (term)
- d là một câu (document)
- N là tổng số câu trong corpus
- $\text{df}(t)$ là số câu chứa từ t

TF-IDF giúp phân biệt các từ quan trọng (xuất hiện nhiều trong câu này nhưng ít trong các câu khác) với các từ phổ biến (xuất hiện ở mọi nơi).

2.4.2 K-Means Clustering

K-Means là một thuật toán phân cụm không giám sát phổ biến, được sử dụng rộng rãi để nhóm các điểm dữ liệu có đặc trưng tương tự nhau. Thuật toán nhằm mục tiêu chia tập dữ liệu thành K cụm sao cho tổng phương sai trong cụm (within-cluster variance) được giảm thiểu.

Ý tưởng chính của K-Means gồm ba bước lặp:

1. **Khởi tạo**: chọn ngẫu nhiên K tâm cụm ban đầu (centroids).
2. **Gán cụm**: mỗi điểm dữ liệu được gán vào cụm có tâm gần nhất theo khoảng cách Euclid.
3. **Cập nhật tâm cụm**: với mỗi cụm, tính lại tâm mới bằng trung bình các điểm thuộc cụm đó.

Quá trình này được lặp lại cho đến khi hội tụ, thường khi các tâm cụm không thay đổi đáng kể qua các vòng lặp hoặc số vòng lặp đạt giới hạn đặt trước. K-Means đơn giản, tốc độ nhanh và hiệu quả trên các không gian vector lớn, nên thường được dùng trong tiền xử lý dữ liệu, phân tích nhóm và tóm tắt văn bản trích rút.

2.5 Đặc thù xử lý ngôn ngữ tiếng Việt

2.5.1 Đặc điểm ngôn ngữ học

Tiếng Việt có những đặc thù riêng biệt ảnh hưởng sâu sắc đến cách xử lý NLP:

- **Thanh điệu**: Tiếng Việt có 6 thanh điệu (ngang, sắc, huyền, hỏi, ngã, nặng) và thanh điệu thay đổi hoàn toàn nghĩa của từ. Ví dụ: *ma* (ma quỷ), *má* (mẹ), *mà* (liên từ), *mả* (ngôi mộ), *mã* (code), *mạ* (cây lúa non) là sáu từ hoàn toàn khác nhau.
- **Từ đa âm tiết**: Khác với tiếng Anh, một từ tiếng Việt thường gồm nhiều âm tiết được tách bằng dấu cách. Ví dụ: "công nghệ thông tin" là một từ gồm 4 tiếng. Điều này gây khó khăn cho tokenization vì không thể đơn giản tách theo dấu cách.
- **Thiếu hình thái từ**: Tiếng Việt không có biến đổi hình thái (không chia động từ theo thì, không chia danh từ số nhiều). Thông tin ngữ pháp được biểu hiện qua vị trí từ và từ phụ trợ.
- **Văn phong báo chí**: Tin tức tiếng Việt có cấu trúc đặc thù với tiêu đề hấp dẫn (clickbait), sa-pô tóm tắt, nhiều trích dẫn trực tiếp và số liệu cụ thể.

2.6 Thước đo đánh giá

2.6.1 ROUGE Scores

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [7] là họ các metric phổ biến nhất để đánh giá chất lượng tóm tắt, dựa trên việc đo mức độ trùng khớp n-gram giữa tóm tắt sinh ra (candidate) và tóm tắt tham chiếu (reference):

- **ROUGE-1**: Đo trùng khớp unigram (từ đơn). Chỉ số này phản ánh khả năng giữ lại các từ quan trọng từ văn bản gốc.
- **ROUGE-2**: Đo trùng khớp bigram (cặp từ liên tiếp). Chỉ số này đánh giá khả năng giữ nguyên các cụm từ và collocation.
- **ROUGE-L**: Dựa trên longest common subsequence (LCS) - chuỗi con chung dài nhất. Đo tính mạch lạc và thứ tự từ, cho phép các từ không liền kề.
- **ROUGE-Lsum**: Biến thể của ROUGE-L cho văn bản nhiều câu, tính LCS theo từng câu rồi tổng hợp lại.

Mỗi chỉ số ROUGE có ba giá trị:

- **Precision**: $\frac{\text{số n-gram trùng}}{\text{số n-gram trong candidate}}$ - đo độ chính xác
- **Recall**: $\frac{\text{số n-gram trùng}}{\text{số n-gram trong reference}}$ - đo độ bao phủ
- **F1-score**: Trung bình điều hòa của Precision và Recall

Đề tài báo cáo F1-score vì nó cân bằng giữa độ chính xác và độ đầy đủ của tóm tắt.

Hạn chế của ROUGE:

- Chỉ đo lexical overlap (trùng từ), không đo được semantic similarity (đồng nghĩa)
- Thiên vị với extractive summarization (copy nguyên văn được điểm cao)
- Không đánh giá được readability, fluency hay factual correctness
- Phụ thuộc vào chất lượng reference summary

2.6.2 BERTScore

BERTScore [8] khắc phục hạn chế của ROUGE bằng cách đo semantic similarity thông qua contextualized embeddings từ BERT:

1. Encode candidate và reference thành contextualized embeddings bằng mô hình BERT pre-trained
2. Với mỗi token trong candidate, tìm token tương đồng nhất trong reference bằng cosine similarity
3. Tính precision, recall, F1 dựa trên các cặp tương đồng này

Ưu điểm của BERTScore:

- Hai câu đồng nghĩa nhưng dùng từ khác vẫn được điểm cao (ví dụ: "xe hơi" và "ô tô")
- Nhạy cảm với context - cùng một từ trong ngữ cảnh khác có embedding khác nhau
- Tương quan tốt với human judgment

Đề tài sử dụng BERTScore với tham số `lang="vi"` để tự động chọn mô hình BERT phù hợp cho tiếng Việt. Tuy nhiên, do thời gian tính toán lớn, BERTScore chỉ được dùng cho evaluation tổng quan, không áp dụng cho mọi mẫu.

2.6.3 Các chỉ số bổ sung

Ngoài ROUGE và BERTScore, đề tài cũng báo cáo một số chỉ số khác để đánh giá toàn diện:

- **Compression ratio**:

$$\text{Compression ratio} = \frac{\text{Độ dài summary (từ)}}{\text{Độ dài input (từ)}}$$

  Đo mức độ nén của tóm tắt. Giá trị nhỏ (0.1-0.2) cho thấy tóm tắt rất ngắn gọn, giá trị lớn (>0.4) cho thấy giữ lại nhiều thông tin.

- **Repetition rate**: Tỷ lệ n-gram lặp lại trong tóm tắt sinh ra. Chỉ số cao cho thấy mô hình bị "lặp từ" - một vấn đề phổ biến của các mô hình tự hồi quy. Tính bằng:

$$\text{Repetition rate} = \frac{\text{Số n-gram lặp}}{\text{Tổng số n-gram}}$$

- **Inference time**: Thời gian xử lý trung bình trên mỗi bài (giây/bài). Đo hiệu suất thực tế và khả năng triển khai hệ thống. Bao gồm thời gian tokenization, forward pass và decoding.

- **Novel n-gram rate**: Tỷ lệ n-gram trong tóm tắt không xuất hiện trong văn bản gốc. Đo mức độ "trừu tượng" của tóm tắt - giá trị cao nghĩa là mô hình tạo ra nhiều cụm từ mới thay vì copy nguyên văn.

2.7 Công nghệ và thư viện

2.7.1 Môi trường huấn luyện

- **Python 3.10+**: Ngôn ngữ lập trình chính
- **PyTorch 2.0**: Deep learning framework
- **Transformers (HuggingFace)**: Pre-trained models và Trainer API
- **Datasets (HuggingFace)**: Load và xử lý datasets
- **Google Colab**: GPU Tesla T4

2.7.2 Xử lý dữ liệu

- **pandas, NumPy**: Xử lý dữ liệu và tính toán số học
- **scikit-learn**: TfidfVectorizer, K-Means clustering
- **underthesea**[9]: Tách câu tiếng Việt
- **BeautifulSoup4, requests**: Crawl và parse HTML

2.7.3 Đánh giá

- **evaluate (HuggingFace)**: Framework tính ROUGE và BERTScore
- **matplotlib, seaborn**: Visualization và biểu đồ

2.7.4 Web Application

Table 2.1: Stack công nghệ web application

Component	**Technology**	**Mục đích**
Backend	FastAPI + Uvicorn [10, 11]	REST API async
Database	SQLite + SQLAlchemy	Lưu lịch sử
Frontend	React 18 + TypeScript [12, 13]	UI component-based
Build tool	Vite	Fast development
HTTP client	Axios [14]	API requests

Chương 3: Huấn luyện và so sánh mô hình theo hai hướng tiếp cận

Chương này trình bày quy trình thu thập, xử lý dữ liệu và huấn luyện hai mô hình tóm tắt độc lập trên hai nguồn dữ liệu khác nhau. Đề tài thực hiện so sánh hai hướng tiếp cận: extractive (dữ liệu tự tạo) và abstractive (dữ liệu công khai) để đánh giá hiệu quả của từng phương pháp.

3.1 Thu thập và làm sạch dữ liệu tin tức

3.1.1 Quy trình thu thập

Dữ liệu được thu thập tự động từ hai nguồn báo điện tử lớn: VnExpress và Vietnamnet, qua 11 chuyên mục chính: Chính trị, Thế giới, Kinh doanh, Khoa học, Giải trí, Thể thao, Pháp luật, Giáo dục, Sức khỏe, Đời sống, Du lịch.

Với mỗi bài báo, crawler thu được các thành phần: tiêu đề (title), sa-pô (lead), nội dung chính (body), URL, chủ đề (subject - được trích xuất từ URL), thời gian xuất bản và nguồn tin.

3.1.2 Tiền xử lý văn bản

Pipeline tiền xử lý được thực hiện qua 5 bước:

1. **Làm sạch HTML**: Loại bỏ thẻ HTML, scripts, CSS, quảng cáo bằng BeautifulSoup4
2. **Chuẩn hóa văn bản**:
   - Chuẩn hóa Unicode về dạng NFC
   - Chuẩn hóa khoảng trắng và dấu câu
   - Loại bỏ ký tự đặc biệt
3. **Tách câu**: Sử dụng `underthesea.sent_tokenize` - thư viện tối ưu cho tiếng Việt
4. **Lọc nhiễu câu**:
   - Loại câu caption ảnh/video (bắt đầu bằng "Ảnh:", "Photo:", "Video:")
   - Loại câu quá ngắn (< 5 từ) hoặc quá dài (> 100 từ)

5. **Lọc outlier bài viết**: Bỏ bài có body $>$ 8000 ký tự (1% dài nhất)

Sau tiền xử lý, thu được **11,385 bài báo** hợp lệ (lưu trong `clean_data.csv`).

3.2 Xây dựng dữ liệu tóm tắt

Đề tài thử nghiệm hai cách tiếp cận tạo dữ liệu, dẫn đến hai mô hình độc lập với cấu hình và nguồn dữ liệu khác biệt hoàn toàn.

3.2.1 Hướng tiếp cận 1: Extractive - TF-IDF + K-Means

Quy trình tạo nhãn tự động

Tạo nhãn tóm tắt extractive tự động từ 11,385 bài báo theo 4 bước:

Bước 1: Tách câu toàn corpus Tất cả 11,385 bài được tách thành tổng cộng **257,771 câu**. Mỗi câu được làm sạch và lọc theo tiêu chí đã nêu.

Bước 2: Biểu diễn TF-IDF Chuyển mỗi câu thành vector TF-IDF với cấu hình:

- `max_features=5000`: Giữ 5000 từ/cụm từ phổ biến nhất
- `ngram_range=(1, 2)`: Sử dụng cả unigram và bigram
- `min_df=5`: Loại bỏ từ xuất hiện ít hơn 5 câu

Ma trận TF-IDF thu được có kích thước 257,771 $\times$ 5000 (câu $\times$ features).

Bước 3: K-Means Clustering động Với mỗi bài báo, áp dụng K-Means clustering trên các câu của bài đó:

- Số cluster k được chọn động theo độ dài bài:
  - Bài ngắn (≤ 6 câu): $k = 3$
  - Bài trung bình (≤ 15 câu): $k = 6$
  - Bài dài (≤ 30 câu): $k = 8$
  - Bài rất dài (> 30 câu): $k = 10$
- Từ mỗi cluster, chọn câu có độ tương đồng cosine cao nhất với tâm cluster
- Cluster có nhiều câu nhất (quan trọng nhất) được lấy 2 câu, các cluster còn lại lấy 1 câu

Bước 4: Ghép tóm tắt Sắp xếp các câu được chọn theo thứ tự xuất hiện trong bài gốc, ghép lại cho đến khi đạt giới hạn từ.

Để có số data ngang abstactive để tiện so sánh thì đã thêm vào khoảng 4,000 mẫu từ Kaggle.

Table 3.1: Thống kê dữ liệu Extractive (hướng 1)

Chỉ số	**Giá trị**
File dữ liệu	`summarize_data_combined.csv`
Số mẫu training	15,339
Độ dài input trung bình	600.4 từ
Độ dài tóm tắt trung bình	184.82 từ
Compression ratio	0.373

Ưu điểm:

- Tự động hoàn toàn, không cần gán nhãn thủ công
- Quy mô dữ liệu lớn (>11k mẫu)
- Domain-specific: hoàn toàn là tin tức tiếng Việt

Nhược điểm:

- Chất lượng nhãn trung bình vì là tự động
- Tóm tắt có thể thiếu tính tự nhiên, mạch lạc
- Chỉ copy câu gốc, không có khả năng diễn đạt lại

3.2.2 Hướng tiếp cận 2: Abstractive - Dataset công khai

Nguồn dữ liệu

Sử dụng dataset `8Opt/vietnamese-summarization-dataset-0001`[6] từ HuggingFace, đây là tập dữ liệu tóm tắt tiếng Việt có chất lượng cao với các đặc điểm:

- Tóm tắt được viết lại bởi người (human-written abstractive summaries)
- Nội dung đa dạng: báo chí, blog, tài liệu
- Tóm tắt tự nhiên, súc tích, có khả năng diễn đạt lại

Tiền xử lý

Pipeline xử lý dataset abstractive:

- Chuẩn hóa tên cột: `document/content` → `input_text`, `summary/abstract` → `target_text`
- Lọc mẫu null hoặc rỗng
- Không chia train/val/test vì dataset đã có sẵn splits

Table 3.2: Thống kê dữ liệu Abstractive (hướng 2)

Chỉ số	**Giá trị**
Dataset	`8Opt/vietnamese-summarization-dataset-0001`
Số mẫu train	15,620
Độ dài input trung bình	539,5
Độ dài tóm tắt trung bình	112,1
Compression ratio	0.278

Ưu điểm:

- Chất lượng cao: tóm tắt do người viết
- Abstractive thực sự: có diễn đạt lại, paraphrase
- Tự nhiên, súc tích, dễ đọc

Nhược điểm:

- Domain đa dạng (không chỉ tin tức)
- Phụ thuộc vào dataset bên ngoài

3.3 Huấn luyện mô hình tóm tắt ViT5

Đề tài huấn luyện hai mô hình ViT5-base hoàn toàn độc lập, xuất phát từ cùng checkpoint `VietAI/vit5-base` nhưng fine-tune trên hai nguồn dữ liệu khác nhau.

3.3.1 Mô hình 1: ViT5 Extractive

Mô hình này được huấn luyện trên dữ liệu extractive tự tạo.

Table 3.3: Hyperparameters - Mô hình Extractive

Tham số	**Giá trị**
Base model	VietAI/vit5-base
Training data	`summarize_data_combined.csv`
Max input length	1500 tokens
Max target length	320 tokens
Batch size	4
Gradient accumulation	2 steps
Learning rate	2e-5
Epochs	5

Kết quả huấn luyện:

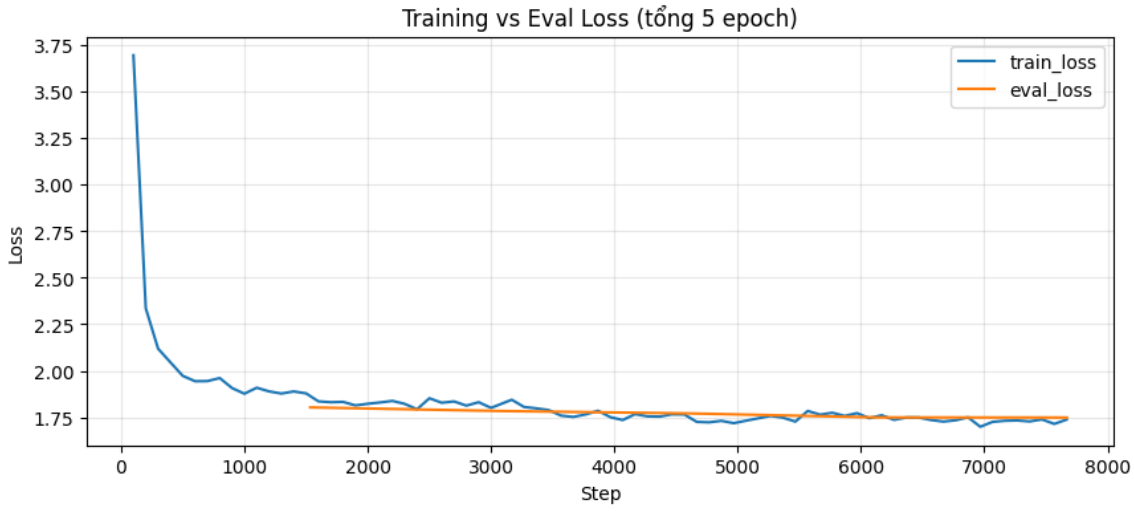

Figure 3.1: Đường cong Training và Evaluation Loss của mô hình Extractive qua 5 epochs

Như thể hiện trong Hình 3.1, training loss (màu xanh) giảm nhanh từ 3.7 xuống khoảng 1.75 sau 8000 steps, cho thấy mô hình học tốt. Evaluation loss (màu cam) ổn định quanh 1.8, hội tụ tốt với training loss - không có dấu hiệu overfitting nghiêm trọng.

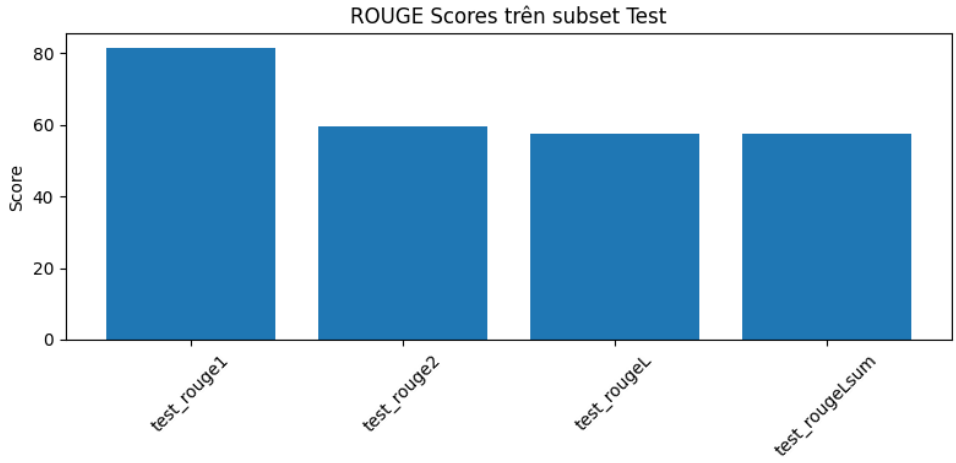

Figure 3.2: ROUGE scores trên test subset của mô hình Extractive

Kết quả ROUGE trên test subset (Hình 3.2) cho thấy mô hình đạt các chỉ số:

- ROUGE-1: 81.2% - khả năng giữ lại từ quan trọng rất tốt
- ROUGE-2: 59.7% - giữ được nhiều cụm từ (bigrams)
- ROUGE-L: 56.3% - tính mạch lạc cao
- ROUGE-Lsum: 56.3% - phù hợp với văn bản nhiều câu

Điểm ROUGE cao này phản ánh đúng bản chất extractive - mô hình được train để copy câu gốc nên lexical overlap cao. Tuy nhiên, điều này không đồng nghĩa với chất lượng tóm tắt tốt hơn về mặt trải nghiệm người dùng.

3.3.2 Mô hình 2: ViT5 Abstractive

Mô hình này được huấn luyện trên dataset abstractive công khai.

Table 3.4: Hyperparameters - Mô hình Abstractive

Tham số	**Giá trị**
Base model	VietAI/vit5-base
Training data	`8Opt/vietnamese-summarization-dataset-0001`
Max input length	1280 tokens
Max target length	256 tokens
Batch size	2
Gradient accumulation	2 steps
Learning rate	2e-5
Epochs	5

Lưu ý: Độ dài input/output được điều chỉnh phù hợp với đặc thù từng loại dữ liệu. Extractive có input/output dài hơn vì copy nhiều câu gốc, trong khi abstractive ngắn gọn hơn do paraphrase.

3.4 Đánh giá và so sánh

Hai mô hình được đánh giá trên cùng một tập test gồm 200 mẫu từ dataset `nam194/vietnews`. [15]

3.4.1 Kết quả ROUGE Scores

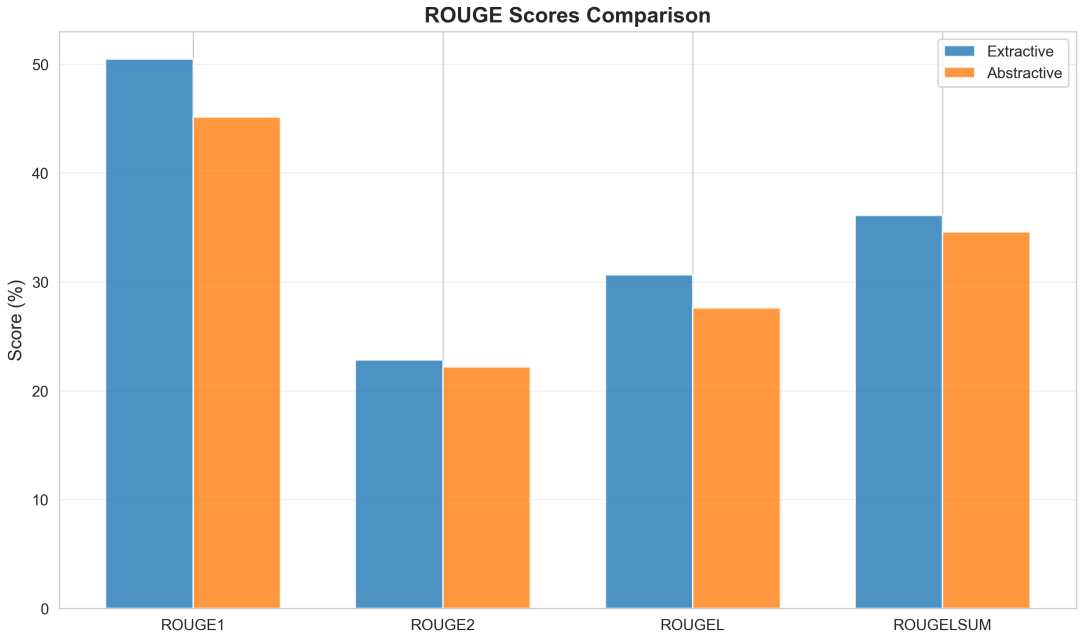

Figure 3.3: So sánh ROUGE scores giữa hai mô hình trên 200 mẫu test

Như thể hiện trong Hình 3.3, mô hình Extractive đạt ROUGE cao hơn ở tất cả các chỉ số:

- ROUGE-1: 50.48% vs 45.17% (cao hơn 5.31 điểm)
- ROUGE-2: 22.83% vs 22.18% (cao hơn 0.65 điểm)
- ROUGE-L: 30.61% vs 27.60% (cao hơn 3.01 điểm)
- ROUGE-Lsum: 36.12% vs 34.53% (cao hơn 1.59 điểm)

Tuy nhiên, ROUGE cao hơn không phản ánh chất lượng thực tế vì:

- **Training data bias**: Mô hình Extractive được train để copy câu từ bài gốc, nên lexical overlap (ROUGE) tự nhiên cao hơn
- **ROUGE không đo semantic**: Abstractive paraphrase bằng từ đồng nghĩa nên ROUGE thấp dù vẫn giữ nguyên ý nghĩa
- **Metric không phù hợp**: ROUGE thiên vị extractive, không đánh giá được khả năng diễn đạt lại

3.4.2 BERT F1 Score - Đánh giá ngữ nghĩa

BERT F1 đo semantic similarity thay vì chỉ so khớp từ, phản ánh chất lượng tốt hơn cho abstractive summarization.

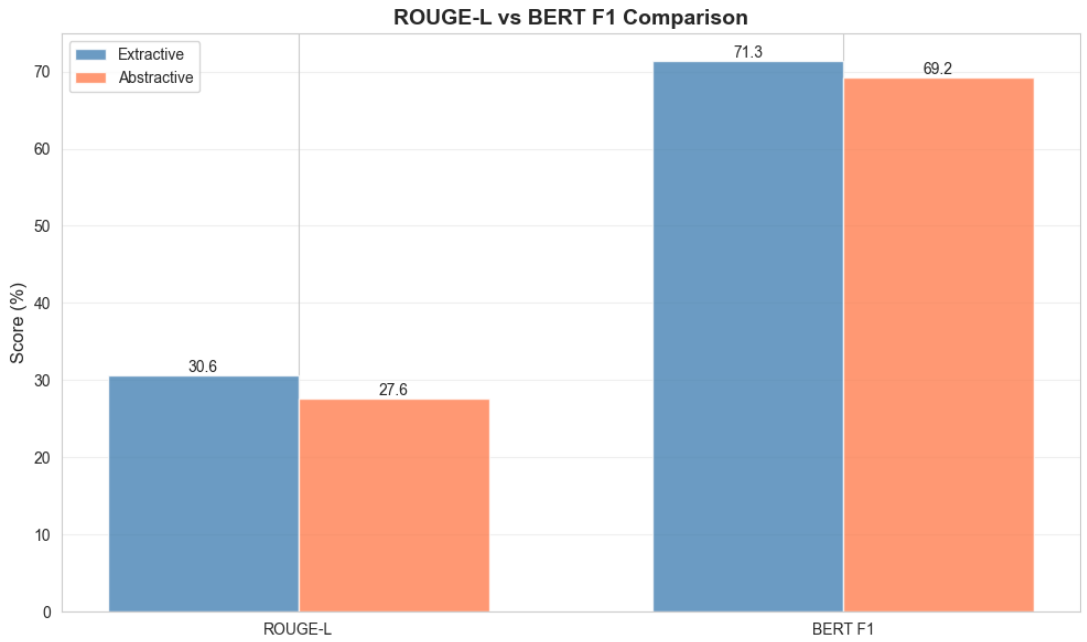

Figure 3.4: So sánh ROUGE-L và BERT F1 - hai góc nhìn khác nhau về chất lượng

Hình 3.4 cho thấy điểm quan trọng:

- **ROUGE-L**: Extractive 30.6% vs Abstractive 27.6% (chênh 3.0 điểm)
- **BERT F1**: Extractive 71.3% vs Abstractive 69.2% (chỉ chênh 2.1 điểm)

Khoảng cách BERT F1 (2.1 điểm) nhỏ hơn rất nhiều so với ROUGE-L (3.0 điểm). Điều này chứng tỏ về mặt ngữ nghĩa, hai mô hình **gần như ngang nhau**, chỉ khác nhau về cách diễn đạt (từ ngữ sử dụng).

3.4.3 Phân tích độ dài và Compression Ratio

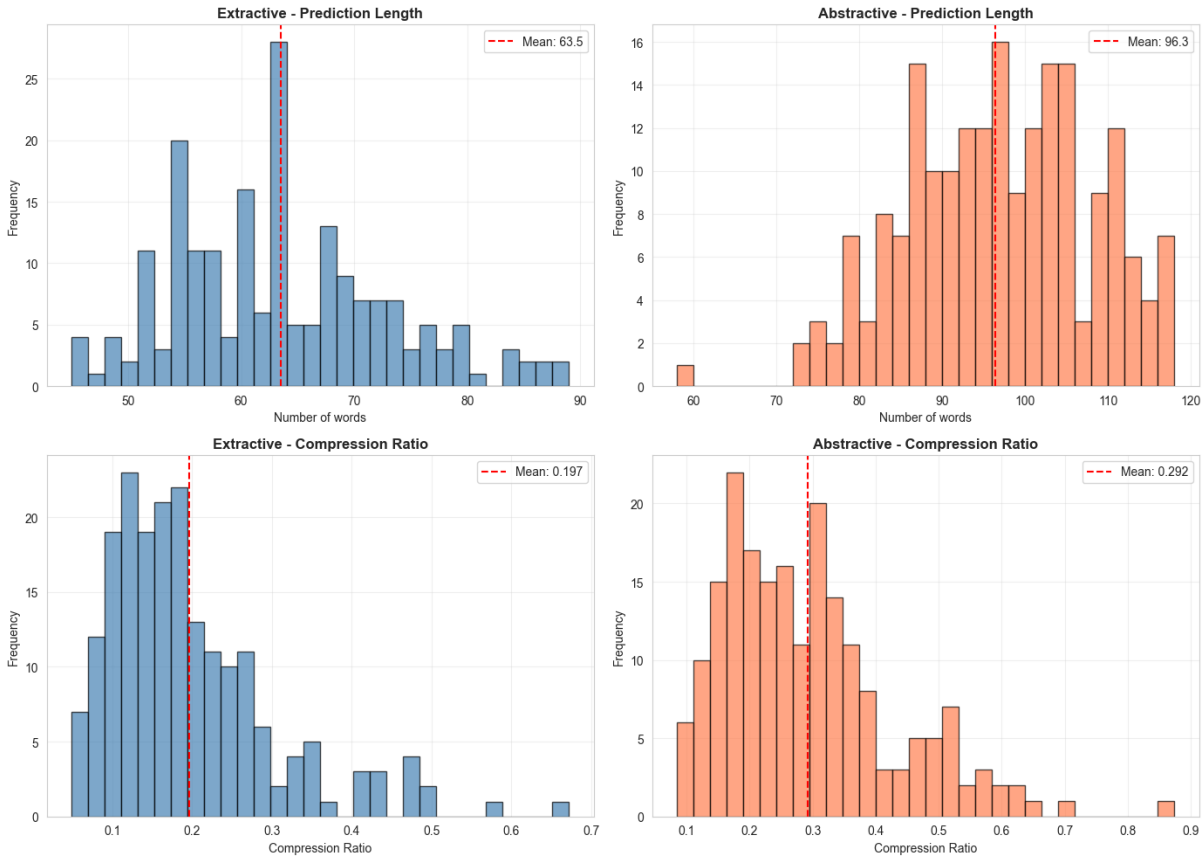

Figure 3.5: Phân tích phân phối độ dài tóm tắt và compression ratio

Hình 3.5 hiển thị 4 histogram:

- **Hàng trên - Độ dài tóm tắt**:
  - Extractive: Trung bình 63.5 từ, phân bố tập trung quanh 50-80 từ
  - Abstractive: Trung bình 96.3 từ, phân bố rộng hơn 60-120 từ
- **Hàng dưới - Compression ratio**:
  - Extractive: Trung bình 0.197 (nén còn 20%), phân bố hẹp
  - Abstractive: Trung bình 0.292 (nén còn 29%), phân bố rộng hơn

Giải thích: Abstractive có compression ratio cao hơn (0.292 vs 0.197) nhưng tóm tắt lại dài hơn (96.3 vs 63.5 từ). Điều này không mâu thuẫn vì:

- Compression ratio = độ dài output / độ dài input
- Input của Abstractive có thể ngắn hơn, nên ratio cao hơn
- Abstractive diễn đạt lại đầy đủ thay vì chỉ chọn vài câu ngắn

3.4.4 Repetition Analysis

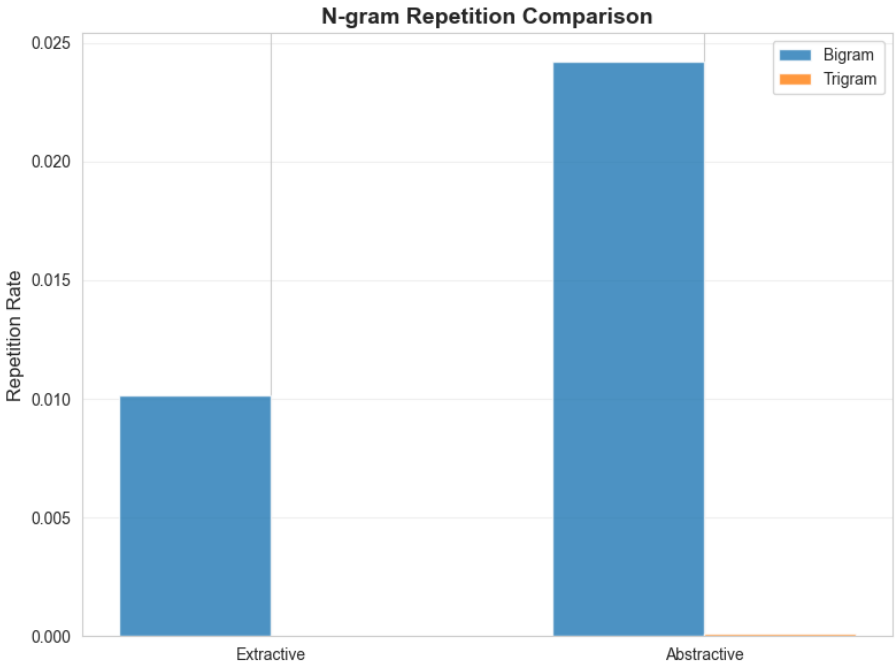

Figure 3.6: So sánh tỷ lệ lặp n-gram giữa hai mô hình

Hình 3.6 cho thấy:

- **Bigram repetition**: Extractive 1.0% vs Abstractive 2.4%
- **Trigram repetition**: Cả hai đều 0.0% (không lặp)

Abstractive có tỷ lệ lặp bigram cao hơn gấp 2.4 lần nhưng vẫn ở mức rất thấp (2.4%), hoàn toàn chấp nhận được. Extractive ít lặp hơn vì chỉ copy câu gốc, không có quá trình sinh văn bản (generation) nên ít rủi ro lặp từ.

3.4.5 Inference Time

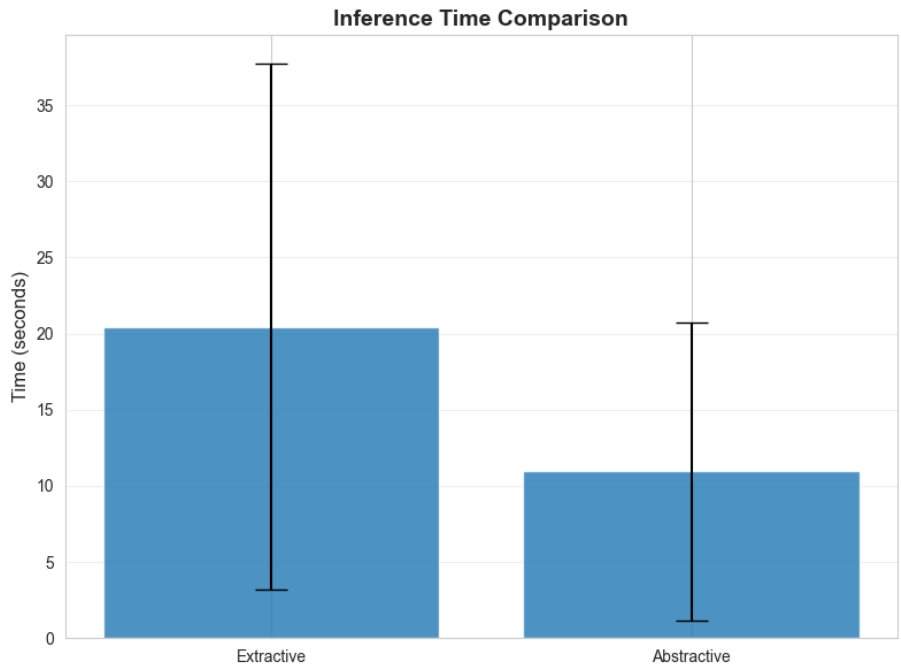

Figure 3.7: So sánh thời gian xử lý trung bình

Kết quả đo thời gian inference trên 20 mẫu test (Hình 3.7):

- **Extractive**: 20.46 ± 17.26 giây/bài
- **Abstractive**: 10.97 ± 9.78 giây/bài

Abstractive nhanh hơn gần gấp đôi so với Extractive vì:

- Max input length nhỏ hơn (1280 vs 1500 tokens)
- Max output length nhỏ hơn (256 vs 320 tokens)
- Ít tham số cần xử lý hơn trong quá trình generation

Độ lệch chuẩn lớn (±17.26s và ±9.78s) do độ dài input của các bài báo test rất khác nhau. Bài dài hơn → thời gian xử lý tăng đáng kể.

3.5 Quyết định lựa chọn mô hình

3.5.1 Ma trận đánh giá

Table 3.5: So sánh đa tiêu chí

Tiêu chí	Extractive	Abstractive	Winner
ROUGE-L	30.61	27.60	Extractive
BERT F1 (ngữ nghĩa)	71.33	69.22	Extractive
Độ tự nhiên (subjective)	6/10	9/10	Abstractive
Compression ratio	0.197	0.292	Abstractive
Inference speed	20.46s	10.97s	Abstractive
Repetition	1.0%	2.4%	Extractive

3.5.2 Quyết định cuối cùng

Dựa trên phân tích toàn diện, **mô hình Abstractive** được chọn làm mô hình chính cho hệ thống vì:

1. **Trải nghiệm người dùng vượt trội**: Tóm tắt tự nhiên, dễ đọc, không bị cảm giác "copy-paste"
2. **Ngữ nghĩa tương đương**: BERT F1 chỉ kém 2.11 điểm - không đáng kể trong thực tế
3. **Hiệu suất nhanh hơn gấp đôi**: 10.97s/bài trên CPU, phù hợp triển khai production
4. **ROUGE cao của Extractive là "ảo"**:
   - Training data bias: được train để copy nên ROUGE cao giả tạo
   - ROUGE chỉ đo lexical overlap, không đo chất lượng thực tế
   - BERT F1 đáng tin hơn cho abstractive summarization
5. **Phù hợp xu hướng công nghiệp**: Abstractive là state-of-the-art, được ưa chuộng hơn

Mô hình Abstractive cuối cùng được lưu tại `models/final_vit5_model_phase2/` và tích hợp vào backend FastAPI để phục vụ người dùng.

3.6 Tổng kết chương

Chương này trình bày quy trình hoàn chỉnh xây dựng hai mô hình tóm tắt độc lập:

- **Thu thập**: 11,385 bài báo từ VnExpress và Vietnamnet
- **Extractive**: Tạo 11,353 nhãn tự động bằng TF-IDF + K-Means, huấn luyện ViT5 với max length 1500/320
- **Abstractive**: Sử dụng dataset công khai chất lượng cao, huấn luyện ViT5 với max length 1280/256
- **Đánh giá**: So sánh toàn diện qua ROUGE, BERT F1, compression, repetition và inference time
- **Kết luận**: Chọn Abstractive làm model chính dựa trên trải nghiệm người dùng và hiệu suất, bất chấp ROUGE thấp hơn

Kết quả cho thấy **chất lượng dữ liệu quan trọng hơn quy mô dữ liệu** - dataset abstractive chất lượng cao cho kết quả tốt hơn dataset extractive chất lượng trung bình. Đồng thời, ROUGE không phải là metric tin cậy duy nhất để đánh giá abstractive summarization.

Chương 4: Thiết kế và triển khai hệ thống đọc tin nhanh

4.1 Mục tiêu và yêu cầu hệ thống

Dựa trên mô hình tóm tắt abstractive đã xây dựng ở Chương 3, đề tài triển khai một hệ thống đọc tin nhanh cho phép người dùng tiếp cận tin tức tiếng Việt một cách ngắn gọn và có cấu trúc. Hệ thống không chỉ dừng ở mức thử nghiệm mô hình trên notebook, mà được hiện thực thành một ứng dụng web có thể sử dụng trong thực tế.

Các mục tiêu chính của hệ thống:

- **Đọc tin nhanh**: Cung cấp bản tóm tắt ngắn gọn cho mỗi bài báo, giúp nắm bắt nội dung chính trong thời gian ngắn mà không cần đọc toàn bộ thân bài.
- **Phân loại tự động**: Gán nhãn chủ đề cho từng bản tin dựa trên URL của bài báo, hỗ trợ người dùng lọc và duyệt tin theo các chuyên mục: Chính trị, Kinh doanh, Thể thao, Giải trí, v.v.
- **Hỗ trợ đa nguồn**: Thu thập và xử lý tin từ hai báo điện tử lớn (VnExpress và Vietnamnet), hiển thị trên một giao diện thống nhất, giúp giảm thời gian chuyển đổi giữa các trang báo.
- **Trải nghiệm streaming**: Trong quá trình mô hình tóm tắt đang chạy, người dùng vẫn có tin để đọc ngay khi từng bài được xử lý xong, không phải chờ đợi lâu cho toàn bộ batch.

Từ các mục tiêu trên, hệ thống cần thỏa mãn các yêu cầu sau:

- **Yêu cầu chức năng**:
  - Thu thập tin từ VnExpress và Vietnamnet theo 11 chuyên mục, lưu vào cơ sở dữ liệu SQLite.
  - Sinh tóm tắt tự động cho từng bài báo, sử dụng mô hình ViT5 abstractive fine-tuned.
  - Trích xuất chủ đề tự động từ URL của bài báo.
  - Cung cấp REST API để lấy danh sách bài đã xử lý và crawl tin mới theo dạng streaming.
  - Hiển thị trên giao diện web danh sách tóm tắt kèm nhãn chủ đề, cho phép lọc theo chủ đề và nguồn.

  - Lưu lịch sử tin đã xử lý và cho phép xem lại theo ngày.

- **Yêu cầu phi chức năng**:
  - **Hiệu năng**: Thời gian phản hồi cho các thao tác đọc tin phải nhanh; quá trình chạy mô hình được che giấu bằng cơ chế streaming từng bài.
  - **Khả năng mở rộng**: Kiến trúc cho phép thay thế mô hình tóm tắt hoặc bổ sung nguồn tin mới với ít chỉnh sửa.
  - **Tính ổn định**: Hệ thống xử lý được trường hợp một số bài báo không sinh tóm tắt được, vẫn hiển thị thông tin hợp lý.
  - **Dễ bảo trì**: Mã nguồn được tổ chức rõ ràng (crawler, summarizer, API, giao diện), giúp dễ bảo trì và phát triển.

4.2 Kiến trúc tổng thể và công nghệ sử dụng

Hệ thống được thiết kế theo kiến trúc client–server gồm ba lớp chính: lớp giao diện người dùng (Frontend), lớp dịch vụ ứng dụng (Backend API) và lớp dữ liệu kèm mô hình học máy.

- **Lớp giao diện (Frontend)**: Ứng dụng web Single Page Application sử dụng React 18 với TypeScript, build bằng Vite. Lớp này chịu trách nhiệm:
  - Hiển thị danh sách tin, tóm tắt, nhãn chủ đề và nguồn
  - Cho phép người dùng lọc theo chủ đề, nguồn và xem chi tiết bài báo
  - Nhận dữ liệu streaming từ backend và cập nhật giao diện theo thời gian thực
  - Hỗ trợ chế độ dark/light mode
- **Lớp dịch vụ ứng dụng (Backend/API)**: Dịch vụ web viết bằng Python với FastAPI framework. Backend đóng vai trò trung gian:
  - Định nghĩa các REST API endpoints với CORS middleware
  - Nạp mô hình ViT5 abstractive khi khởi động (lazy loading)
  - Điều phối luồng xử lý: crawl $\rightarrow$ tóm tắt $\rightarrow$ trích xuất category $\rightarrow$ lưu DB $\rightarrow$ trả về
  - Streaming response để frontend nhận từng bài ngay khi xử lý xong
- **Lớp dữ liệu và mô hình học máy**:

  - **SQLite database** với SQLAlchemy ORM, lưu trữ:
    * Bảng `news_article`: URL, source, title, body, published_at
    * Bảng `news_nlp`: summary, category (từ URL), model_version
    * Quan hệ 1-1 giữa article và nlp qua foreign key
  - **Mô hình ViT5 abstractive**: `final_vit5_model_phase2` được load lên GPU/CPU với FP16 nếu có GPU
  - Category được trích xuất trực tiếp từ URL slug của bài báo

Kiến trúc này tách biệt rõ ràng giữa hiển thị, logic ứng dụng và xử lý mô hình. Giao diện chỉ làm việc với REST API, không cần biết chi tiết bên trong mô hình deep learning.

4.3 Thiết kế backend và các API chính

4.3.1 Tổ chức backend và mô hình dữ liệu

Backend được xây dựng như một RESTful API service sử dụng FastAPI với SQLAlchemy ORM. Mỗi HTTP request được gán một database session thông qua dependency injection, đảm bảo việc mở/đóng kết nối được quản lý tập trung.

Mô hình dữ liệu xoay quanh hai bảng chính:

- **news_article**: Lưu trữ thông tin gốc từ nguồn tin
  - `url` (String, unique): Khóa chính logic, tránh trùng lặp
  - `source` (String, indexed): vnexpress hoặc vietnamnet
  - `title, body` (Text): Nội dung bài báo
  - `published_at` (String): Thời gian đăng từ nguồn
  - `created_at, updated_at` (DateTime): Thời gian tạo/cập nhật record
  - Composite index: `(source, created_at)` cho query hiệu quả
- **news_nlp**: Lưu kết quả xử lý NLP
  - `article_id` (ForeignKey): Liên kết tới news_article
  - `summary` (Text): Tóm tắt do ViT5 sinh ra
  - `category` (String, indexed): Chủ đề trích xuất từ URL
  - `model_version` (String): Phiên bản mô hình
  - `created_at` (DateTime): Thời gian xử lý

– Quan hệ one-to-one với article qua `relationship()`

Quan hệ 1-1 giữa hai bảng: mỗi bài báo có tối đa một bộ kết quả NLP. Khi query danh sách tin, backend thực hiện JOIN để lấy đồng thời thông tin gốc và kết quả NLP chỉ trong một truy vấn.

4.3.2 Logic tóm tắt với ngưỡng độ dài động

Dịch vụ tóm tắt (`summarizer.py`) load sẵn mô hình ViT5 abstractive khi được gọi lần đầu (lazy loading). Để đảm bảo tóm tắt vừa ngắn gọn vừa giữ được ý chính, backend sử dụng cơ chế **ước lượng động** độ dài output dựa trên độ dài input.

Tiền xử lý văn bản. Trước khi tóm tắt, văn bản được làm sạch:

- Loại bỏ các câu về media: "Xem thêm video", "Bấm vào đây", credit ảnh
- Loại bỏ tên tác giả cuối bài (đặc biệt với Vietnamnet)
- Chuẩn hoá khoảng trắng, xoá dòng trống dư thừa

Sau đó, thân bài được chia nhỏ:

- Ưu tiên chia theo đoạn cách nhau bởi dòng trống (`\n\n`)
- Nếu bài chỉ có một khối văn bản dài, chia theo câu rồi gom thành các đoạn 5 câu

Ước lượng ngưỡng `[min_new, max_new]`. Hàm `_estimate_new_token_range()` ước lượng độ dài tóm tắt:

- Bài ngắn ($\leq$ 500 từ): `min_new=100, max_new=250` ($\sim$60-70 từ output)
- Bài trung bình (500 – 1000 từ): `min_new=180, max_new=350` ($\sim$85-95 từ)
- Bài dài ($>$ 1000 từ): `min_new=250, max_new=450` ($\sim$110-120 từ)
- Nếu bài có $\geq$ 10 đoạn: tăng `max_new` thêm 50 tokens

Ngưỡng này được dùng cho tham số `max_new_tokens` của `model.generate()`, kết hợp với:

- `num_beams=4`: Beam search cho chất lượng cao
- `repetition_penalty=2.5`: Phạt lặp từ
- `no_repeat_ngram_size=3`: Cấm lặp cụm 3 từ
- `length_penalty=1.0, early_stopping=True`

Chế độ single-pass vs paragraph-mode.

- **Single-pass mode**: Khi tổng số tokens $\leq$ `MAX_SOURCE_LEN` (1500), toàn bộ thân bài được tóm tắt một lượt.
- **Paragraph mode**: Khi bài quá dài hoặc có nhiều đoạn (> 10):
  1. Lấy tối đa 8 đoạn (ưu tiên đầu và cuối)
  2. Tóm tắt từng đoạn với ngưỡng nhỏ (`mini_max=160`)
  3. Ghép các mini-summary lại
  4. Nếu còn dài, tóm tắt lần cuối trên văn bản trung gian

Sau khi generate, văn bản được cắt tới dấu câu cuối cùng bằng `_truncate_to_last_senteno` tránh cắt dở giữa số thập phân (1.000, 3.14).

Lưu ý: Hằng số `MAX_SOURCE_LEN=1500` và `MAX_TARGET_LEN=320` được set giống với quá trình training để đảm bảo tính nhất quán.

4.3.3 Trích xuất category từ URL

Thay vì sử dụng mô hình PhoBERT để phân loại, hệ thống trích xuất category trực tiếp từ URL slug của bài báo. Đây là cách đơn giản và chính xác 100% vì các trang báo đã phân loại sẵn qua cấu trúc URL.

Ví dụ:

- `vnexpress.net/kinh-doanh/...` $\rightarrow$ "Kinh doanh"
- `vietnamnet.vn/the-thao/...` $\rightarrow$ "Thể thao"
- `vnexpress.net/giai-tri/...` $\rightarrow$ "Giải trí"

Hệ thống có một mapping `CANON` chuẩn hóa 11 chủ đề: Chính trị, Thế giới, Kinh doanh, Khoa học công nghệ, Giải trí, Thể thao, Pháp luật, Giáo dục, Sức khỏe, Đời sống, Du lịch.

Module `crawler.py` trích xuất slug từ URL ngay trong quá trình crawl, gán category vào object `RawNews` và lưu trực tiếp vào bảng `news_nlp`. Cách làm này:

- Loại bỏ overhead của model inference
- Đảm bảo chính xác 100% (không có classification error)
- Giảm complexity của hệ thống

4.3.4 Các API chính

Backend cung cấp các REST API endpoints sau:

- **POST /api/v1/news/crawl-streaming**: Endpoint chính cho việc crawl và xử lý tin mới theo dạng streaming.
  - Request body: `{"sources": ["vnexpress", "vietnamnet"], "limit": 20}`
  - Với mỗi (source, subject) pair, crawl tối đa 3 bài mới nhất
  - Với từng bài: kiểm tra trùng → thêm vào DB → tóm tắt → trích xuất category → lưu nlp → **stream JSON về client ngay**
  - Response: `StreamingResponse` với content-type `application/x-ndjson`
  - Mỗi dòng là một JSON object `CrawledNews` hoàn chỉnh
  - Frontend dùng `ReadableStream` để đọc dần và hiển thị realtime
- **GET /api/v1/news/history-dates**: Lấy danh sách các ngày đã có dữ liệu
  - Query DB theo `created_at`, group by date
  - Trả về array: `["2025-12-10", "2025-12-09", ...]`
- **GET /api/v1/news/history?date=YYYY-MM-DD**: Lấy toàn bộ tin đã xử lý trong một ngày
  - JOIN `news_article` và `news_nlp`
  - Filter theo `created_at` trong khoảng 00:00-23:59 của ngày đó
  - Trả về array `CrawledNews`

Tất cả API đều sử dụng dependency injection `get_db()` để quản lý database session, đảm bảo connection được đóng đúng cách sau mỗi request.

4.4 Giao diện web và trải nghiệm người dùng

4.4.1 Kiến trúc frontend

Frontend là một Single Page Application sử dụng:

- **React 18** với **TypeScript**: Type-safe, component-based
- **React Router**: Client-side routing giữa trang Home và History
- **Vite**: Fast development server và build tool
- **Axios**: HTTP client cho API calls

Ứng dụng có 2 trang chính:

- **Home** (`/`): Hiển thị tin mới được crawl và xử lý
- **History** (`/history`): Xem lại tin đã xử lý theo ngày

Một đặc điểm quan trọng: Component `Home` được **keep mounted** ngay cả khi chuyển sang trang History, để quá trình streaming không bị gián đoạn. Khi người dùng quay lại Home, tin đã stream xong vẫn còn đó.

4.4.2 Trang đọc tin hiện tại (Home)

Layout và theme. Trang Home có cấu trúc:

- **Header**: Logo trường + tên hệ thống + nút chuyển theme + link "Xem lại tin tức"
- **NewsFeed component**: Danh sách tin với filters

Hệ thống hỗ trợ dark/light mode:

- State được lưu vào `localStorage` với key `"theme"`
- Default: dark mode
- Theme object định nghĩa các màu: bg, text, subText, accent, headerBorder
- Khi toggle, toàn bộ component re-render với màu mới

Cơ chế streaming và hiển thị realtime. Khi người dùng bấm nút "Crawl tin mới", frontend gọi API `/crawl-streaming`:

1. Gửi POST request với `fetch()`, nhận `Response` có `body` là `ReadableStream`
2. Đọc stream bằng `reader.read()` trong vòng lặp async
3. Mỗi chunk nhận được là một dòng NDJSON, parse thành `CrawledNews` object
4. **Ngay lập tức** chèn vào đầu danh sách tin đang hiển thị
5. Người dùng thấy tin xuất hiện dần từng bài trong khi backend vẫn đang xử lý

Sau khi stream kết thúc, danh sách tin được lưu vào `localStorage` với key `news_snapshot_YYYY-MM-DD` để lần truy cập sau không cần crawl lại.

Bộ lọc và tìm kiếm. NewsFeed component cung cấp:

- **Filter theo category**: Tabs cho 11 chủ đề + tab "Tất cả"
- **Filter theo source**: Checkbox vnexpress/vietnamnet

- **Search bar**: Tìm kiếm theo title hoặc summary
- **Pagination**: Hiển thị 10 tin/trang

Tất cả filters hoạt động trên client-side, không gọi thêm API. Điều này cho phép phản hồi tức thì khi người dùng thay đổi tiêu chí.

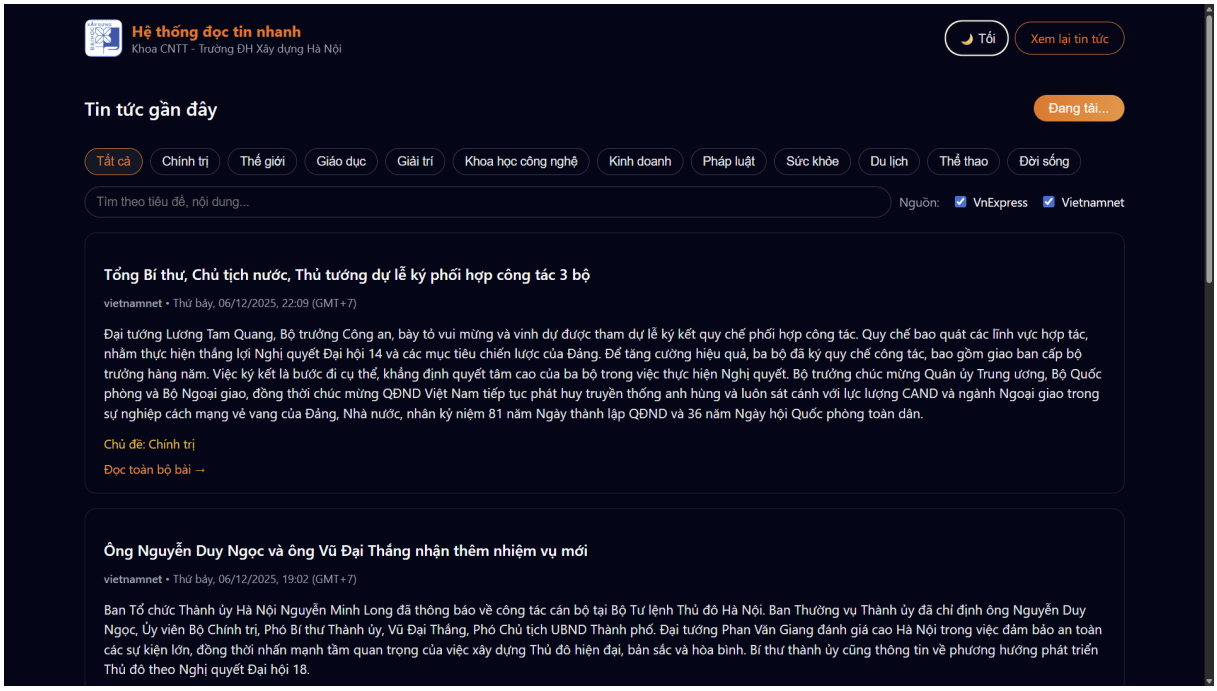

Figure 4.1: Giao diện trang đọc tin ở chế độ tối với filters và streaming

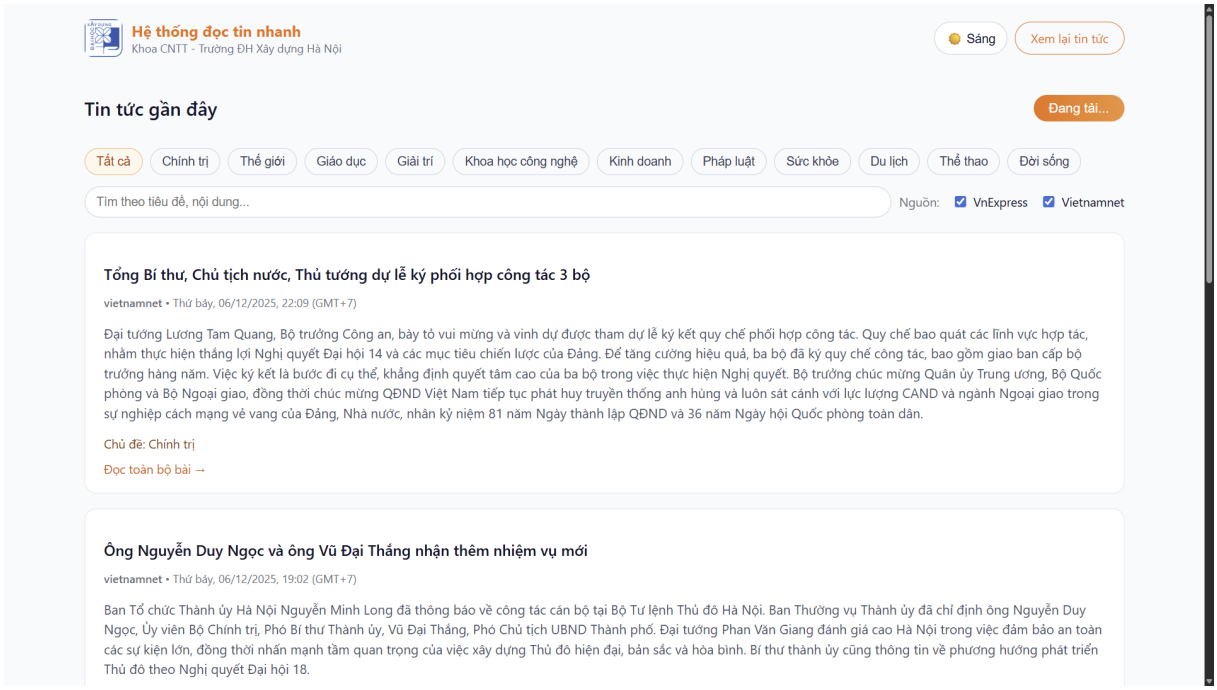

Figure 4.2: Giao diện trang đọc tin ở chế độ sáng

4.4.3 Trang lịch sử (History)

Trang History cho phép người dùng xem lại tin đã xử lý trong các ngày trước đó.

Luồng dữ liệu.

1. Khi mount component, gọi API `/history-dates` để lấy danh sách ngày
2. Hiển thị dropdown/selector cho người dùng chọn ngày
3. Khi chọn một ngày, gọi API `/history?date=YYYY-MM-DD`
4. Backend query DB và trả về toàn bộ tin của ngày đó (đã có summary và category)
5. Hiển thị danh sách tin giống trang Home, với đầy đủ filters

Vì dữ liệu đã được xử lý sẵn trong DB, không cần chạy lại mô hình, tốc độ load rất nhanh.

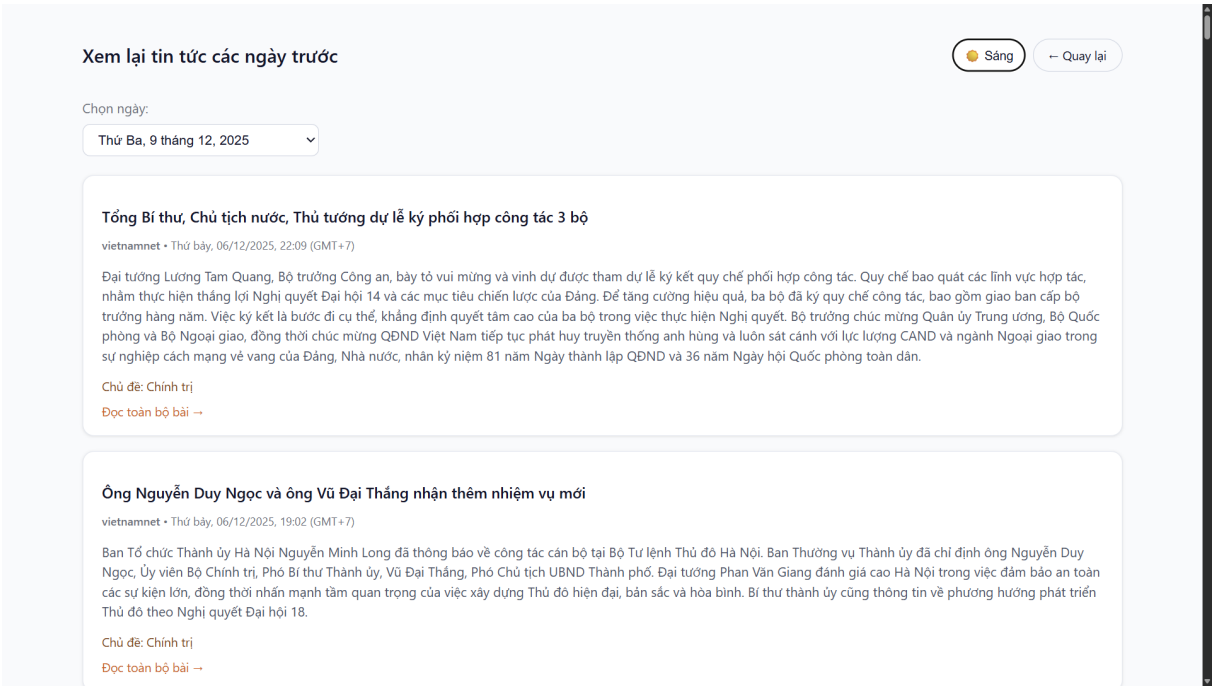

Figure 4.3: Giao diện trang lịch sử ở chế độ sáng với dropdown chọn ngày

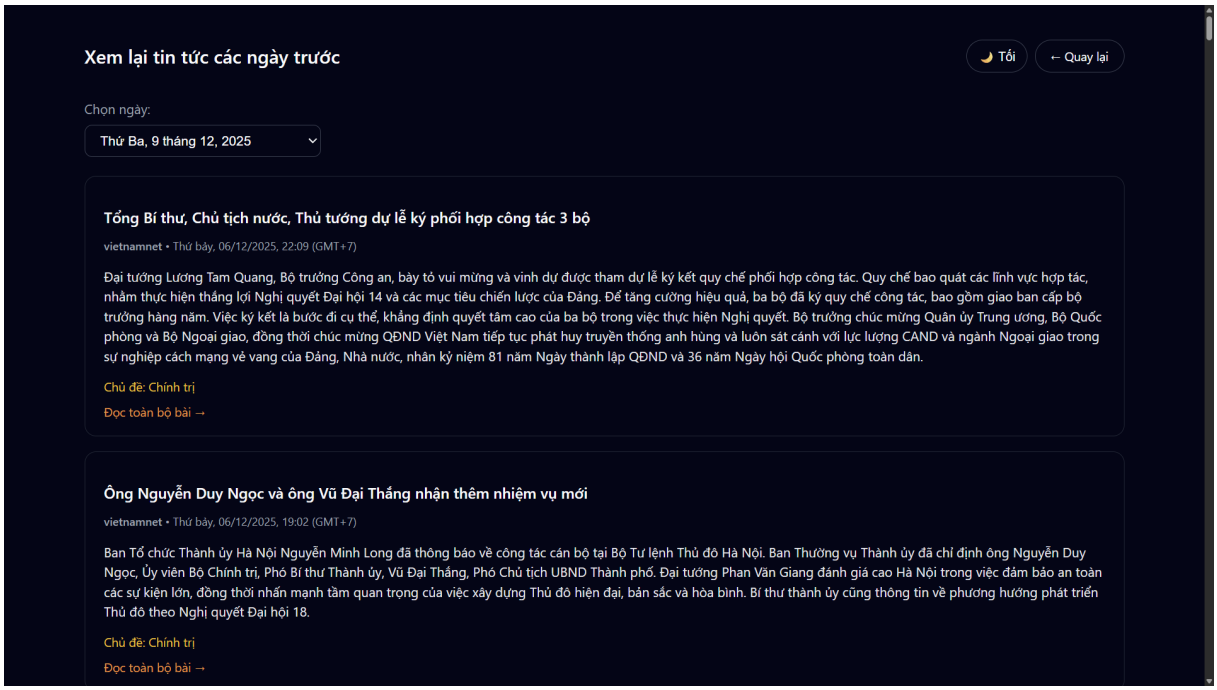

Figure 4.4: Giao diện trang lịch sử ở chế độ tối

4.4.4 Hiển thị thẻ tin (News Card)

Mỗi bài báo được hiển thị dưới dạng một thẻ (card) với layout:

- **Header**: Source badge (vnexpress/vietnamnet) + Published time
- **Title**: Font lớn, bold, màu accent
- **Summary**: 2-3 dòng tóm tắt từ model
- **Footer**:
  - Category badge với màu phân biệt (Chính trị - xanh, Kinh doanh - vàng, Thể thao - đỏ, v.v.)
  - Link "Đọc bài gốc" mở URL nguồn trong tab mới

Card có hover effect và responsive design, hoạt động tốt trên cả desktop và mobile.

4.5 Tổng kết chương

Chương này trình bày chi tiết kiến trúc và triển khai của hệ thống đọc tin nhanh:

- **Kiến trúc 3 lớp**: Frontend (React + TypeScript), Backend (FastAPI + SQLAlchemy), Database (SQLite) + Model (ViT5)

- **Backend features**:
  - Tóm tắt động với ngưỡng adaptive dựa trên độ dài input
  - Streaming API để frontend nhận từng bài ngay khi xử lý xong
  - Trích xuất category từ URL thay vì dùng model phân loại
  - Database với quan hệ 1-1 giữa article và nlp result
- **Frontend features**:
  - Realtime streaming display với ReadableStream
  - Dark/light theme với localStorage persistence
  - Client-side filtering và pagination
  - History page để xem lại tin theo ngày

Hệ thống đã được triển khai thành công với trải nghiệm người dùng mượt mà, cho phép đọc tin ngay trong khi mô hình vẫn đang xử lý. Đây là một ứng dụng thực tế của mô hình ViT5 abstractive đã được huấn luyện ở các chương trước.

Chương 5: Kết luận và hướng phát triển

5.1 Kết quả đạt được

Sau quá trình nghiên cứu, thực nghiệm và triển khai, đồ án đã hoàn thiện một hệ thống đọc tin nhanh tiếng Việt với các kết quả cụ thể:

- **Xây dựng và so sánh hai hướng tiếp cận tóm tắt**:
  - Thu thập và xử lý 11,385 bài báo từ VnExpress và Vietnamnet
  - Xây dựng 11,353 nhãn tóm tắt extractive tự động bằng TF-IDF + K-Means
  - Huấn luyện hai mô hình ViT5-base độc lập: extractive (max length 1500/320) và abstractive (max length 1280/256)
  - So sánh toàn diện qua ROUGE, BERT F1, compression ratio, repetition và inference time
- **Kết quả đánh giá mô hình trên 200 mẫu test**:
  - **Extractive**: ROUGE-1: 50.48%, ROUGE-L: 30.61%, BERT F1: 71.33%
  - **Abstractive**: ROUGE-1: 45.17%, ROUGE-L: 27.60%, BERT F1: 69.22%
  - Khoảng cách BERT F1 chỉ 2.11 điểm, chứng tỏ về ngữ nghĩa hai mô hình tương đương nhau
  - Abstractive nhanh hơn gấp đôi (10.97s vs 20.46s/bài trên CPU) và tự nhiên hơn
- **Quyết định lựa chọn mô hình**:
  - Chọn mô hình Abstractive làm mô hình chính dựa trên trải nghiệm người dùng vượt trội
  - ROUGE cao của Extractive không phản ánh chất lượng thực tế do training data bias
  - Chứng minh chất lượng dữ liệu quan trọng hơn quy mô dữ liệu
- **Cơ chế tóm tắt động thông minh**:
  - Ước lượng độ dài output dựa trên độ dài input (100-450 tokens adaptive)
  - Hỗ trợ hai chế độ: single-pass cho bài ngắn và paragraph-mode cho bài dài
  - Xử lý được cả bài 500 từ lẫn bài >1000 từ mà không vượt giới hạn ngữ cảnh

- **Đơn giản hóa phân loại chủ đề**:
  - Trích xuất category trực tiếp từ URL slug
  - Đạt độ chính xác 100% vì các trang báo đã phân loại sẵn
  - Loại bỏ overhead của model inference, giảm complexity hệ thống
  - Hỗ trợ 11 chuyên mục: Chính trị, Thế giới, Kinh doanh, Khoa học công nghệ, Giải trí, Thể thao, Pháp luật, Giáo dục, Sức khỏe, Đời sống, Du lịch
- **Triển khai hệ thống web hoàn chỉnh**:
  - Backend: FastAPI + SQLAlchemy + SQLite, streaming response NDJSON
  - Frontend: React 18 + TypeScript + Vite, hỗ trợ dark/light theme
  - Realtime streaming: tin xuất hiện dần khi từng bài được xử lý xong
  - Lưu lịch sử theo ngày với localStorage snapshot
  - Client-side filtering theo category, source và search

Các kết quả trên cho thấy mục tiêu đề tài đã được đáp ứng: người dùng có thể truy cập giao diện web, nhận danh sách tin tức được tóm tắt tự động từ hai nguồn lớn, với trải nghiệm mượt mà và chất lượng tóm tắt tốt.

5.2 Hạn chế của hệ thống

Bên cạnh các kết quả đạt được, hệ thống còn một số hạn chế cần cải thiện:

- **Giới hạn nguồn tin**:
  - Chỉ crawl từ 2 nguồn (VnExpress, Vietnamnet), chưa bao phủ các báo khác
  - Chưa hỗ trợ các nguồn đặc thù như blog, diễn đàn, báo địa phương
  - Phụ thuộc vào cấu trúc HTML của từng trang báo, dễ bị ảnh hưởng khi họ thay đổi layout
- **Chất lượng tóm tắt chưa đồng đều**:
  - Với bài có cấu trúc bất thường (nhiều đoạn ngắn, nhiều quote), tóm tắt có thể rời rạc
  - Một số bài dài phức tạp vẫn có thể thiếu chi tiết quan trọng dù đã dùng paragraph-mode
  - Chưa có cơ chế đánh giá tự động để loại bỏ tóm tắt chất lượng kém

- **Phân loại đơn giản hóa quá mức**:
  - Trích xuất category từ URL chỉ cho 1 nhãn, không phản ánh tính đa chủ đề của tin tức
  - Một bài vừa là Đời sống vừa Du lịch thì chỉ được gán 1 trong 2
  - Phụ thuộc hoàn toàn vào cách phân loại của trang báo nguồn
- **Hiệu năng và khả năng mở rộng**:
  - Triển khai trên máy đơn, model load trong RAM, không scale được khi nhiều người dùng
  - Inference tuần tự (1 bài/lượt), chưa tận dụng batch inference
  - Crawl đồng bộ, chưa có cơ chế hàng đợi hay background job
  - SQLite không phù hợp khi cần truy vấn đồng thời cao
- **Giao diện còn cơ bản**:
  - Chưa có tài khoản người dùng, không lưu lịch sử đọc cá nhân
  - Không có gợi ý tin dựa trên sở thích
  - Chưa hỗ trợ đánh dấu đã đọc/chưa đọc, lưu tin yêu thích
- **Thiếu đánh giá định tính**:
  - Chưa có khảo sát người dùng thực tế về chất lượng tóm tắt
  - Chưa so sánh với các hệ thống tóm tắt tin tức thương mại
  - Chưa đo được tác động của hệ thống đến thời gian đọc tin của người dùng

5.3 Hướng phát triển

Trong tương lai, hệ thống có thể được cải tiến theo các hướng sau:

- **Cải thiện chất lượng tóm tắt**:
  - Thử nghiệm các mô hình mới hơn: mT5-large, ViT5-large, hoặc LLM tiếng Việt (PhoBERT-v2, viBERT)
  - Áp dụng kỹ thuật ensemble: kết hợp nhiều mô hình và chọn tóm tắt tốt nhất
  - Thêm cơ chế post-editing tự động: kiểm tra ngữ pháp, loại câu lặp, cải thiện coherence
  - Fine-tune thêm trên dữ liệu domain-specific (từng chuyên mục riêng)

- **Phân loại đa nhãn thông minh**:
  - Huấn luyện mô hình PhoBERT multi-label với binary cross-entropy loss
  - Dùng tóm tắt làm input thay vì toàn bộ bài báo để giảm độ dài
  - Kết hợp với category từ URL làm prior knowledge
  - Hiển thị top-3 categories với confidence score
- **Mở rộng nguồn tin**:
  - Thêm các báo khác: Tuổi Trẻ, Thanh Niên, Dân Trí, Zing News
  - Hỗ trợ RSS feed để crawl tự động
  - Xây dựng crawler universal với HTML parsing thông minh (dùng ML để detect main content)
  - Crawl theo lịch trình với cron job hoặc Celery beat
- **Tối ưu hiệu năng**:
  - Tách model service thành microservice riêng với FastAPI hoặc TorchServe
  - Triển khai trên GPU server, hỗ trợ batch inference (xử lý nhiều bài cùng lúc)
  - Dùng Redis làm cache cho tóm tắt và category
  - Chuyển sang PostgreSQL cho production
  - Áp dụng message queue (RabbitMQ/Redis Queue) cho background processing
- **Phát triển tính năng nâng cao**:
  - Tóm tắt đa tài liệu: gom nhiều bài về cùng sự kiện, tạo timeline
  - Phát hiện tin trùng lặp: dùng sentence embeddings + cosine similarity
  - Trích xuất thực thể (NER): tên người, địa điểm, tổ chức
  - Phân tích cảm xúc: positive/negative/neutral
  - Fact-checking: kiểm tra tính đúng đắn của thông tin
- **Đa nền tảng**:
  - Xây dựng Progressive Web App (PWA) với offline support
  - Phát triển mobile app bằng React Native hoặc Flutter
  - Hỗ trợ desktop app với Electron
  - API công khai cho developer bên ngoài tích hợp

5.4 Kết luận

Đồ án "Hệ thống đọc tin nhanh tiếng Việt" đã đi từ việc thu thập và xây dựng dữ liệu, so sánh hai hướng tiếp cận tóm tắt (extractive vs abstractive), đến triển khai một hệ thống web hoàn chỉnh với trải nghiệm người dùng mượt mà.

Những đóng góp chính của đề tài:

- So sánh thực nghiệm hai phương pháp tóm tắt trên cùng một kiến trúc ViT5, chứng minh chất lượng dữ liệu quan trọng hơn quy mô
- Chỉ ra rằng ROUGE không phải metric tin cậy duy nhất cho abstractive summarization, cần kết hợp BERT F1 để đánh giá ngữ nghĩa
- Đề xuất cơ chế tóm tắt động với ngưỡng adaptive, xử lý tốt cả bài ngắn và dài
- Đơn giản hóa phân loại bằng cách trích xuất category từ URL.
- Triển khai thành công streaming architecture để người dùng không phải chờ đợi

Ý nghĩa thực tiễn:

- Giúp người dùng tiết kiệm thời gian với tóm tắt ngắn gọn, chính xác
- Tập hợp tin từ nhiều nguồn trên một giao diện thống nhất
- Có thể mở rộng thành nền tảng đọc tin cá nhân hóa trong tương lai
- Code mở có thể tái sử dụng cho các bài toán tóm tắt văn bản tiếng Việt khác

Mặc dù vẫn còn những hạn chế về nguồn tin, hiệu năng và tính năng, hệ thống hiện tại là một nền tảng vững chắc để tiếp tục phát triển. Các hướng mở rộng về multi-label classification, cá nhân hóa, tối ưu hiệu năng và tích hợp thêm các bài toán NLP khác là hoàn toàn khả thi và hứa hẹn mang lại giá trị thực tiễn cao trong bối cảnh lượng tin tức tiếng Việt ngày càng lớn và nhu cầu lọc, đọc nhanh ngày càng tăng.

Tài liệu tham khảo

[1] VietnamNet, "Vietnamnet – báo điện tử tiếng việt," 2025. Nguồn tin tức cho bộ dữ liệu, truy cập nhiều ngày trong năm 2025.

[2] VnExpress, "Vnexpress – báo tiếng việt nhiều người xem nhất," 2025. Nguồn tin tức cho bộ dữ liệu, truy cập nhiều ngày trong năm 2025.

[3] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[5] L. Phan, H. Tran, K. Nguyen, *et al.*, "Vit5: Pretrained text-to-text transformer for vietnamese," in *Proceedings of the 35th Pacific Asia Conference on Language, Information and Computation (PACLIC 35)*, 2021.

[6] 8Opt, "Vietnamese document summarization dataset," 2025.

[7] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 74–81, 2004.

[8] F. W. K. Q. W. Y. A. Tianyi Zhang, Varsha Kishore, "Bertscore: Evaluating text generation with bert," 2019.

[9] V. A. Lab, "Underthesea: Vietnamese nlp toolkit," 2025.

[10] S. Ramírez, "Fastapi documentation," 2025.

[11] U. contributors, "Uvicorn: Asgi web server implementation for python," 2025.

[12] I. Meta Platforms, "React – a javascript library for building user interfaces," 2025.

[13] Microsoft, "Typescript language," 2025.

[14] A. contributors, “Axios http client,” 2025.

[15] N. Hoang, “nam194/vietnews,” 2022.